Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.00017

ĐỀ XUẤT MỘT SỐ THUẬT TOÁN HEURISTIC GIẢI BÀI TOÁN CÂY STEINER NHỎ NHẤT

Trần Việt Chương[1], Phan Tấn Quốc[2], Hà Hải Nam[3]

[1] Trung tâm Công nghệ thông tin và Truyền thông, Sở Thông tin và Truyền thông tỉnh Cà Mau

[2] Khoa Công nghệ thông tin, Trường Đại học Sài Gòn

[3]Viện Khoa học Kỹ thuật Bưu điện, Học viện Công nghệ Bưu chính viễn thông

chuongtv@camau.gov.vn, quocpt@sgu.edu.vn, namhh@ptit.edu.vn

***TÓM TẮT**: Cây Steiner nhỏ nhất (Steiner Minimal Tree - SMT) là bài toán tối ưu tổ hợp có nhiều ứng dụng quan trọng trong khoa học và kỹ thuật; đây là bài toán thuộc lớp NP-hard. Trong hàng chục năm qua, đã có nhiều bài báo khoa học công bố cách giải bài toán SMT theo các hướng tiếp cận giải chính xác và giải gần đúng. Trong bài báo này, hai thuật toán heuristic giải bài toán SMT được đề xuất; đề xuất thứ nhất là sử dụng ý tưởng chính của thuật toán tìm cây đường đi ngắn nhất và đề xuất thứ hai là kết hợp ý tưởng chính của các thuật toán Prim và Dijkstra. Các thuật toán đề xuất đã được thực nghiệm trên các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn; kết quả thực nghiệm cho thấy rằng các thuật toán đề xuất là hiệu quả hơn một số thuật toán giải gần đúng hiện biết; nhất là đối với các đồ thị thưa có kích thước lớn. Các kết quả thực nghiệm này là thông tin hữu ích cho các nghiên cứu tiếp theo về bài toán SMT.*

***Từ khóa**: Steiner minimal tree, sparse graph, heuristic algorithm, metaheuristic algorithm.*

I. GIỚI THIỆU

A. Một số định nghĩa

Mục này trình bày một số định nghĩa và tính chất liên quan đến bài toán cây Steiner nhỏ nhất.

Định nghĩa 1. *Cây Steiner* [3]

Cho $G = (V(G), E(G))$ là một đơn đồ thị vô hướng liên thông và có trọng số không âm trên cạnh; trong đó $V(G)$ là tập gồm n đỉnh, $E(G)$ là tập gồm m cạnh, $w(e)$ là trọng số của cạnh e, $e \in E(G)$. Cho L là tập con các đỉnh của $V(G)$, cây T đi qua tất cả các đỉnh trong L được gọi là cây Steiner của L.

Tập L được gọi là tập *terminal*, các đỉnh thuộc tập L được gọi là đỉnh *terminal*, đỉnh thuộc cây T mà không thuộc tập L được gọi là đỉnh Steiner. Khác với các bài toán cây khung thường gặp, cây Steiner chỉ cần đi qua tất cả các đỉnh thuộc tập *terminal* L và có thể thêm một số đỉnh khác nữa thuộc tập $V(G)$.

Định nghĩa 2. Chi phí cây Steiner [3]

Cho $T = (V(T), E(T))$ là một cây Steiner của đồ thị G, chi phí của cây T, ký hiệu là $C(T)$, là tổng trọng số của các cạnh thuộc cây T, tức là $C(T) = \sum_{e \in E(T)} w(e)$.

Định nghĩa 3. *Cây Steiner nhỏ nhất* [3]

Cho đồ thị G được mô tả như trên, bài toán tìm cây Steiner có chi phí nhỏ nhất được gọi là bài toán cây Steiner nhỏ nhất (Steiner Minimal Trees problem – *SMT*); hoặc cũng có thể được gọi ngắn gọn hơn đó là bài toán cây Steiner Steiner Trees problem).

SMT là bài toán tối ưu tổ hợp trên lý thuyết đồ thị. Trong trường hợp tổng quát, *SMT* đã được chứng minh thuộc lớp bài toán *NP-hard* [3,16]. Có hai trường hợp đặc biệt đối với bài toán *SMT* là giải được trong thời gian đa thức; đó là khi $L=V(G)$ và khi $|L|=2$ ($|L|$ là ký hiệu số lượng đỉnh của tập Y): Khi $L=V$ thì bài toán *SMT* có thể giải bằng các thuật toán tìm cây khung nhỏ nhất; chẳng hạn như các thuật toán Prim [20], Kruskal [20]; khi $|L|=2$ thì bài toán *SMT* có thể giải được bằng các thuật toán tìm đường đi ngắn nhất giữa hai đỉnh; chẳng hạn như thuật toán Dijkstra [20].

Để ngắn gọn, trong bài báo này từ *đồ thị* được hiểu là đơn đồ thị, vô hướng, liên thông, có trọng số không âm.

B. Ứng dụng của bài toán SMT

Bài toán *SMT* có thể được tìm thấy trong các ứng dụng quan trọng trong một số lĩnh vực khoa học và kỹ thuật; chẳng hạn như bài toán thiết kế mạng truyền thông [17,19], bài toán thiết kế VLSI (Very Large Scale Integrated) [9,12], các bài toán liên quan đến hệ thống mạng với chi phí nhỏ nhất [2, 4, 8, 29],… Do là bài toán thuộc lớp *NP-hard* nên ứng dụng của bài toán *SMT* cần được xem xét dưới các góc độ của bài toán thiết kế hay là dưới góc độ của bài toán thực thi: Nếu ở góc độ thiết kế thì ưu tiên hơn về chất lượng lời giải hơn; nếu ở góc độ thực thi thì ưu tiên về thời gian chạy hơn.

C. Một số nghiên cứu liên quan bài toán SMT và vấn đề đặt ra cần giải quyết

Do có tính khoa học và tính ứng dụng rộng rãi, bài toán *SMT* đã thu hút được sự quan tâm nghiên cứu của nhiều nhà khoa học trên thế giới trong hàng chục năm qua; đã có hàng loạt thuật toán giải bài toán *SMT* được đề xuất và có thể chia chúng làm các hướng tiếp cận sau:

Hướng thứ nhất là các thuật toán tìm lời giải đúng. Chẳng hạn thuật toán quy hoạch động của Dreyfus và Wagner [23], thuật toán dựa trên phép nới lỏng Lagrange của Beasley [11], thuật toán nhánh cận của Koch và Martin [27],… Ưu điểm của hướng tiếp cận này là tìm được lời giải chính xác, nhược điểm của hướng tiếp cận này là chỉ giải được các bài toán kích thước nhỏ [6, 7, 30]. Hướng tiếp cận này là cơ sở quan trọng để đánh giá mức độ chính xác của các thuật toán giải gần đúng. Việc tìm lời giải đúng cho bài toán *SMT* thực sự là một thách thức lớn trong lý thuyết tối ưu tổ hợp [26, 28].

Hướng thứ hai là các thuật toán tìm lời giải gần đúng cận tỉ lệ. Ưu điểm của các thuật toán này là có sự đảm bảo về mặt toán học theo nghĩa lời giải tìm được gần đúng một cận tỉ lệ α nào đó so với lời giải tối ưu, nhược điểm của thuật toán này là cận tỉ lệ tìm được của nhiều thuật toán đề xuất trong thực tế thường là kém hơn rất nhiều so với chất lượng lời giải tìm được bởi nhiều thuật toán gần đúng khác dựa trên thực nghiệm. Thuật toán *MST-Steiner* có cận tỉ lệ 2 (MST-Minimum Spanning Trees) [3, 4], thuật toán *Zelikovsky-Steiner* có cận tỉ lệ 11/6 [3, 4] là các thuật toán điển hình của hướng tiếp cận này.

Hướng thứ ba là các thuật toán heuristic. Thuật toán heuristic chỉ những kinh nghiệm riêng biệt để tìm kiếm lời giải cho một bài toán tối ưu cụ thể. Thuật toán heuristic thường tìm được lời giải có thể chấp nhận được trong thời gian cho phép nhưng không chắc đó là lời giải tốt nhất [14, 22, 24]. Ưu điểm của các thuật toán heuristic là cho thời gian chạy nhanh; điển hình như thuật toán Distance Network Heuristic của Kou, Markowsky và Berman [15]; thuật toán MST-Steiner của Bang Ye Wu và Kun-Mao Chao [3] cũng được xem là một heuristic giải bài toán *SMT*.

Hướng thứ tư là các thuật toán metaheuristic. Thuật toán metaheuristic sử dụng nhiều heuristic kết hợp với các kỹ thuật phụ trợ nhằm khai phá không gian tìm kiếm; metaheuristic thuộc lớp các thuật toán tìm kiếm tối ưu. Hiện đã có nhiều công trình sử dụng thuật toán metaheuristic giải bài toán *SMT*; chẳng hạn như thuật toán local search [21], thuật toán tìm kiếm tabu [5], thuật toán di truyền [1], thuật toán di truyền song song [19],…. Cho đến hiện tại, hướng tiếp cận metaheuristic cho chất lượng lời giải tốt trong số các thuật toán giải gần đúng; tuy nhiên thời gian thực hiện các thuật toán metaheuristic là chậm hơn nhiều so với các thuật toán heuristic.

Vấn đề đặt ra của chúng tôi trong bài báo này là đề xuất thuật toán heuristic giải bài toán *SMT* cho chất lượng lời giải chấp nhận được và thời gian chạy nhanh hơn so với các thuật toán metaheuristic; vấn đề này càng có ý nghĩa khi xét về gốc độ bài toán thực thi khi áp dụng bài toán cây Steiner vào vấn đề thực tế.

II. ĐỀ XUẤT THUẬT TOÁN HEURISTIC GIẢI BÀI TOÁN *SMT*

Trong bài báo này chúng tôi sẽ đề xuất hai heuristic dựa trên thuật toán tìm đường đi ngắn nhất giữa hai đỉnh để giải bài toán *SMT*; và để so sánh chất lượng của các thuật toán đề xuất, chúng tôi trình bày thuật toán heuristic MST-Steiner của Bang Ye Wu và Kun-Mao Chao [3]. Các thuật toán này có dữ liệu đầu vào là đồ thị $G=(V,E,w)$, tập đỉnh terminal $L \subseteq V$; và dữ liệu đầu ra là chi phí của cây Steiner T.

A. Thuật toán MST-Steiner

Algorithm: MST-Steiner [3]

1: Xây dựng đồ thị đầy đủ G_L của tập *terminal L*; trong đó trọng số của các cạnh là chiều dài của đường đi ngắn nhất nối hai đỉnh đầu mút của cạnh đó;
2: Tìm một cây khung nhỏ nhất của G_L (dùng thuật toán Kruskal hoặc Prim);
3: $T=\varnothing$;
4: **for** (với mỗi cạnh $e=(u,v) \in E(T_L)$) {
5: Tìm một đường đi ngắn nhất P từ đỉnh u đến đỉnh v trên đồ thị G;
6: **if** P chứa ít hơn hai đỉnh trong T thì
7: Chèn P vào T;
8: **else**
9: Cho p_i và p_j là đỉnh đầu, đỉnh cuối đã có trong T; chèn đường đi từ u đến p_i và từ p_j đến v vào T;
10: }
11: Output T.

Thuật toán MST-Steiner có độ phức tạp thời gian tính là $O(n|L|^2)$ [3].

B. Thuật toán SPT-Steiner

Định nghĩa 4. Cây đường đi ngắn nhất [3]

Cho đồ thị *G*, cây đường đi ngắn nhất (Shortest Path Tree - *SPT*) có gốc tại đỉnh *s* là cây khung của *G* với tập cạnh là các cạnh trên các đường đi ngắn nhất xuất phát từ đỉnh *s* đến các đỉnh còn lại của *G*.

Cây đường đi ngắn nhất có gốc tại đỉnh *s* của đồ thị *G* có thể tìm được nhờ áp dụng thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh *s* đến tất cả các đỉnh còn lại.

Algorithm: SPT-Steiner (Shortest Path Tree- Steiner Tree)

1: Tìm các cây đường đi ngắn nhất xuất phát tại các đỉnh thuộc tập *V*: $SPT_1, SPT_2, \ldots, SPT_n$

2: Với mỗi cây $T=SPT_i$, duyệt các đỉnh treo $u \in T$, nếu $u \notin L$ thì xóa cạnh chứa đỉnh *u* khỏi *E*(*T*), xóa đỉnh *u* trong *V*(*T*) và cập nhật bậc của đỉnh kề với đỉnh *u* trong *T*. Lặp lại bước này đến khi *T* không còn thay đổi (bước này gọi là bước xóa các cạnh dư thừa); sau bước này thu được các $SPT_{i'}$ tương ứng với các SPT_i.

3: Tìm một cây $SPT_{i'}$ có tổng trọng số các cạnh là nhỏ nhất;

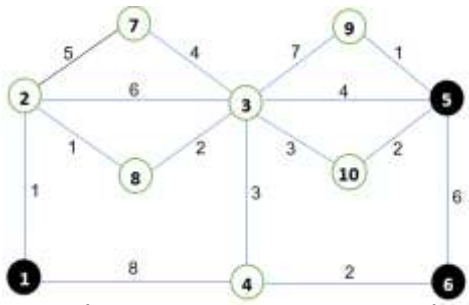
4: Output *T*.

Độ phức tạp thời gian tính cho thuật toán SPT-Steiner

Do thuật toán Dijkstra sử dụng cấu trúc dữ liệu heap có độ phức tạp thời gian tính là $O(m \log n)$ [18,20,25] nên bước 1 có độ phức tạp $O(mn \log n)$; bước 2 có độ phức tạp $O(n^2)$; bước 3 có độ phức tạp $O(n)$. Vậy thuật toán SPT-Steiner có độ phức tạp thời gian tính là $O(mn \log n)$.

Ví dụ 1: Minh họa từng bước thực hiện thuật toán SPT-Steiner

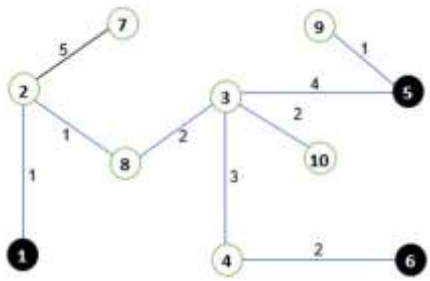
Cho đồ thị *G* có 10 đỉnh và 15 cạnh như hình 1; trong đó tập $L=\{1,5,6\}$.



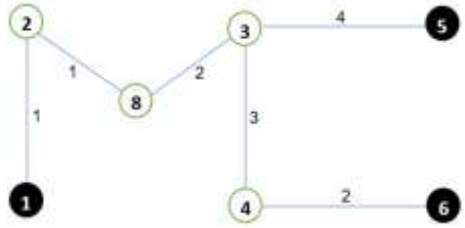
Hình 1. Đồ thị vô hướng liên thông có trọng số G

Theo định nghĩa 4 và thuật toán SPT-Steiner, ta xây dựng các cây đường đi ngắn nhất có gốc tại các đỉnh **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10** và sau khi xóa các cạnh dư thừa ta có các cây Steiner có giá trị lần lượt là: 13, 13, 13, 13, 14, 15, 19, 13, 14, 15; để ngắn gọn, chúng tôi chỉ minh họa việc xây dựng các cây Steiner với gốc xuất phát thuộc tập *terminal*.

Ta xét hình 1; đầu tiên ta tìm cây đường đi ngắn nhất có gốc tại đỉnh **1**; ta thu được cây đường đi ngắn nhất như hình 2.*a*; tiếp theo ta loại bỏ các cạnh dư thừa (**2**,**7**), (**5**,**9**), (**3**,**10**). Kết quả thu được cây Steiner như hình 2.*b* với chi phí là 13.

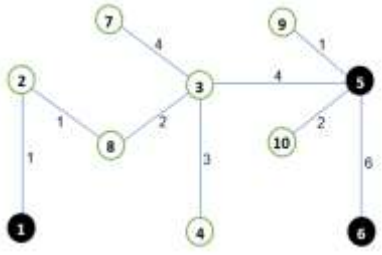


Hình 2.*a*. Cây đường đi ngắn nhất có gốc tại đỉnh **1**

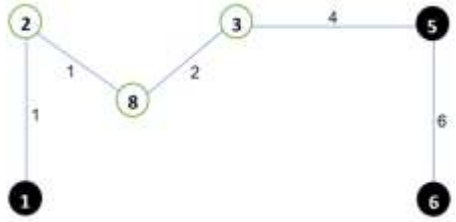


Hình 2.*b*. Cây Steiner sau khi đã xóa cạnh dư thừa

Tương tự, từ hình 1 ta tìm cây đường đi ngắn nhất có gốc tại đỉnh **5**; ta thu được cây đường đi ngắn nhất ở hình 3.*a*; tiếp theo ta loại bỏ các cạnh dư thừa (**3**,**7**), (**5**,**9**), (**5**,**10**), (**3**,**4**). Kết quả thu được cây Steiner như hình 3.*b* với chi phí là 14.

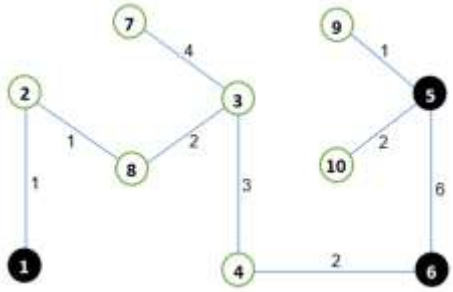


Hình 3.*a*. Cây đường đi ngắn nhất có gốc tại đỉnh **5**

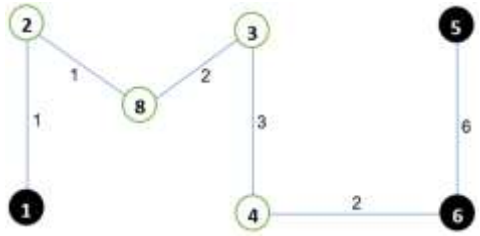


Hình 3.*b*. Cây Steiner sau khi đã xóa cạnh dư thừa

Tương tự, từ hình 1 ta tìm cây đường đi ngắn nhất có gốc tại đỉnh **6**; ta thu được cây đường đi ngắn nhất ở hình 4.*a*; tiếp theo ta loại bỏ các cạnh dư thừa (**3**,**7**), (**5**,**9**), (**5**,**10**). Kết quả thu được cây Steiner như hình 4.*b* với chi phí là 15.



Hình 4.*a*. Cây đường đi ngắn nhất có gốc tại đỉnh **6**



Hình 4.*b*. Cây Steiner sau khi đã xóa cạnh dư thừa

Và theo thuật toán SPT-Steiner trên; cây steiner kết quả tìm được có chi phí là 13; chẳng hạn như hình 2.*b*.

C. Thuật toán PD-Steiner

Algorithm: PD-Steiner (Prim+Dijkstra-Steiner Tree)

1: Tìm các đường đi ngắn nhất từ các đỉnh thuộc tập L đến các đỉnh thuộc tập V;
2: Chọn một đỉnh $u \in L$; đặt $T=\{u\}$;
3: **while** (T chưa chứa tất cả các đỉnh thuộc tập L) {
4: Từ mỗi đỉnh $v \in L$ và v chưa thuộc T, tìm các đường đi ngắn nhất từ đỉnh v đến các đỉnh $z \in T$; và
5: Chọn ra một đường đi ngắn nhất P;
6: Kết nạp các đỉnh và các cạnh trên đường đi P vào cây T;
7: }
8: Output T.

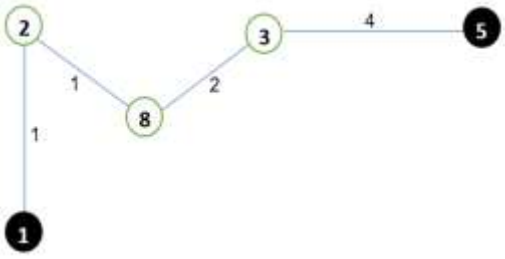
Độ phức tạp thời gian tính cho thuật toán PD-Steiner

Bước 1 có độ phức tạp $O(|L|.m.\log n)$; bước 2 có độ phức tạp $O(1)$; vòng lặp ở bước 3 có độ phức tạp $O(|L|.n^2)$. Vậy thuật toán PD-Steiner có độ phức tạp thời gian tính là $O(|L|.n^2)$.

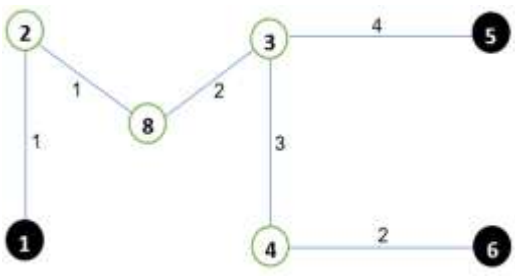
Ví dụ 2: Minh họa từng bước thực hiện thuật toán PD-Steiner

Ta xét hình 1; khoảng cách ngắn nhất từ các đỉnh thuộc tập L={**1**, **5**, **6}** đến các đỉnh thuộc tập V lần lượt là:

Đầu tiên chọn T={**1**}, đường đi ngắn nhất từ đỉnh **5** đến đỉnh **1** như hình 5.*a*; tiếp theo từ tìm đường đi ngắn nhất từ đỉnh **6** đến các đỉnh **1, 2, 3, 5, 8** và ta chọn đường đi ngắn nhất từ đỉnh **6** đến đỉnh **3**; kết quả thu được cây Steiner như hình 5.*b* với chi phí là 13.



Hình 5.*a*. Đường đi ngắn nhất từ đỉnh **1** đến đỉnh **5**



Hình 5.*b*. Đường đi ngắn nhất từ đỉnh **6** đến đỉnh **3**

Và theo thuật toán PD-Steiner trên; cây steiner kết quả tìm được có chi phí là 13; chẳng hạn như hình 5.*b*.

III. THỰC NGHIỆM VÀ ĐÁNH GIÁ

Phần này chúng tôi mô tả chi tiết việc thực nghiệm thuật toán MST-Steiner [3] và hai thuật toán heuristic do chúng tôi đề xuất; đồng thời đưa ra một số đánh giá về các kết quả đạt được.

A. Dữ liệu thực nghiệm

Để thực nghiệm các thuật toán liên quan, chúng tôi sử dụng 78 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn dùng cho bài toán cây Steiner tại địa chỉ URL http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html [10]; trong đó nhóm *steinb* có 18 đồ thị, nhóm *steinc* có 20 đồ thị, nhóm *steind* có 20 đồ thị, nhóm *steine* có 20 đồ thị (các đồ thị này có tối đa 2500 đỉnh). Do cách tiếp cận bài toán *SMT* bằng các thuật toán heuristic có ưu điểm hơn cách tiếp cận bằng các thuật toán metaheuristic đối với các đồ thị có kích thước lớn; ít nhất cũng về mặt thời gian tính; nên chúng tôi đề xuất thêm 20 bộ dữ liệu là các đồ thị thưa kích thước 10000 đỉnh được sinh ngẫu nhiên và đặt tên là *steinf* (các đồ thị này được sinh trước hết từ một cây khung ngẫu nhiên đi qua hết tất cả *n* đỉnh của đồ thị; sau đó sinh ngẫu nhiên thêm một số cạnh đến khi đồ thị có đủ *m* cạnh; trọng số các cạnh cũng được sinh ngẫu nhiên). Tổng cộng hệ thống dữ liệu thực nghiệm trong bài báo này là 98 bộ.

B. Môi trường thực nghiệm

Các thuật toán MST-Steiner, SPT-Steiner, PD-Steiner được chúng tôi cài đặt bằng ngôn ngữ C++ sử dụng môi trường DEV C++ 5.9.2; được thực nghiệm trên một máy chủ ảo Hệ điều hành Windows server 2008 R2 Enterprise, 64bit, Intel(R) Xeon (R) CPU E5-2660 0 @ 2.20 GHz, RAM 4GB.

C. Kết quả thực nghiệm và đánh giá

Kết quả thực nghiệm của các thuật toán được ghi nhận ở các bảng 1, 2, 3, 4, 5. Các bảng có cấu trúc như sau: Cột đầu tiên (Test) là tên các bộ dữ liệu trong hệ thống dữ liệu thực nghiệm; số đỉnh (n), số cạnh (m) và số đỉnh thuộc tập *terminal* ($|L|$) của từng đồ thị; các cột tiếp theo ghi nhận giá trị chi phí cây Steiner tương ứng với từng thuật toán.

Bảng 1. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinb*

Test	*n*	*m*	$\|L\|$	MST-Steiner	SPT-Steiner	PD-Steiner
steinb1.txt	50	63	9	82	82	82
steinb2.txt	50	63	13	90	84	84
steinb3.txt	50	63	25	140	147	138
steinb4.txt	50	100	9	64	59	62
steinb5.txt	50	100	13	64	62	61
steinb6.txt	50	100	25	128	134	126
steinb7.txt	75	94	13	111	111	111
steinb8.txt	75	94	19	104	113	104
steinb9.txt	75	94	38	222	222	220
steinb10.txt	75	150	13	98	90	90
steinb11.txt	75	150	19	91	93	90
steinb12.txt	75	150	38	174	192	174
steinb13.txt	100	125	17	175	172	175
steinb14.txt	100	125	25	237	253	235
steinb15.txt	100	125	50	323	335	318
steinb16.txt	100	200	17	137	138	133
steinb17.txt	100	200	25	134	139	132
steinb18.txt	100	200	50	222	250	222

Bảng 2. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinc*

Test	*n*	*m*	$\|L\|$	MST-Steiner	SPT-Steiner	PD-Steiner
steinc1.txt	500	625	5	88	86	85
steinc2.txt	500	625	10	144	158	144
steinc3.txt	500	625	83	779	843	762
steinc4.txt	500	625	125	1114	1193	1085
steinc5.txt	500	625	250	1599	1706	1583

Test	*n*	*m*	|*L*|	MST-Steiner	SPT-Steiner	PD-Steiner
steinc6.txt	500	1000	5	60	56	55
steinc7.txt	500	1000	10	115	103	102
steinc8.txt	500	1000	83	531	597	516
steinc9.txt	500	1000	125	728	865	718
steinc10.txt	500	1000	250	1117	1327	1107
steinc11.txt	500	2500	5	37	32	34
steinc12.txt	500	2500	10	49	46	48
steinc13.txt	500	2500	83	274	322	268
steinc14.txt	500	2500	125	337	417	332
steinc15.txt	500	2500	250	571	703	562
steinc16.txt	500	12500	5	13	12	12
steinc17.txt	500	12500	10	19	19	20
steinc18.txt	500	12500	83	125	146	123
steinc19.txt	500	12500	125	158	195	159
steinc20.txt	500	12500	250	269	339	268

Bảng 3. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steind*

Test	*n*	*m*	|*L*|	MST-Steiner	SPT-Steiner	PD-Steiner
steind1.txt	1000	1250	5	107	107	107
steind2.txt	1000	1250	10	237	228	232
steind3.txt	1000	1250	167	1636	1771	1593
steind4.txt	1000	1250	250	2012	2174	1957
steind5.txt	1000	1250	500	3310	3511	3270
steind6.txt	1000	2000	5	74	70	75
steind7.txt	1000	2000	10	105	111	103
steind8.txt	1000	2000	167	1138	1287	1104
steind9.txt	1000	2000	250	1540	1773	1500
steind10.txt	1000	2000	500	2163	2550	2141
steind11.txt	1000	5000	5	31	29	31
steind12.txt	1000	5000	10	43	44	42
steind13.txt	1000	5000	167	531	643	518
steind14.txt	1000	5000	250	702	851	691
steind15.txt	1000	5000	500	1151	1437	1134
steind16.txt	1000	25000	5	15	13	14
steind17.txt	1000	25000	10	25	25	23
steind18.txt	1000	25000	167	251	301	246
steind19.txt	1000	25000	250	344	424	334
steind20.txt	1000	25000	500	544	691	542

Bảng 4. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steine*

Test	*n*	*m*	|*L*|	MST-Steiner	SPT-Steiner	PD-Steiner
steine1.txt	2500	3125	5	125	111	115
steine2.txt	2500	3125	10	244	214	227
steine3.txt	2500	3125	417	4232	4570	4118
steine4.txt	2500	3125	625	5316	5675	5201
steine5.txt	2500	3125	1250	8313	8976	8226

Test	*n*	*m*	|*L*|	MST-Steiner	SPT-Steiner	PD-Steiner
steine6.txt	2500	5000	5	86	73	78
steine7.txt	2500	5000	10	165	150	159
steine8.txt	2500	5000	417	2809	3254	2726
steine9.txt	2500	5000	625	3809	4474	3727
steine10.txt	2500	5000	1250	5745	6847	5673
steine11.txt	2500	12500	5	39	34	38
steine12.txt	2500	12500	10	73	68	69
steine13.txt	2500	12500	417	1370	1704	1332
steine14.txt	2500	12500	625	1814	2304	1778
steine15.txt	2500	12500	1250	2856	3626	2819
steine16.txt	2500	62500	5	17	15	15
steine17.txt	2500	62500	10	27	27	26
steine18.txt	2500	62500	417	646	804	639
steine19.txt	2500	62500	625	809	1059	806
steine20.txt	2500	62500	1250	1358	1753	1357

Bảng 5. Kết quả thực nghiệm thuật toán trên nhóm đồ thị *steinf*

Test	*n*	*m*	|*L*|	MST-Steiner	SPT-Steiner	PD-Steiner
steinf1.txt	10000	93750	10	62	58	59
steinf2.txt	10000	93750	20	107	109	105
steinf3.txt	10000	93750	834	2292	2912	2214
steinf4.txt	10000	93750	1250	3022	3865	2921
steinf5.txt	10000	93750	2500	4979	6481	4841
steinf6.txt	10000	125000	10	54	50	50
steinf7.txt	10000	125000	20	92	94	86
steinf8.txt	10000	125000	834	1993	2499	1907
steinf9.txt	10000	125000	1250	2729	3500	2624
steinf10.txt	10000	125000	2500	4328	5641	4228
steinf11.txt	10000	156250	10	37	38	37
steinf12.txt	10000	156250	20	82	84	76
steinf13.txt	10000	156250	834	1816	2313	1717
steinf14.txt	10000	156250	1250	2429	3118	2339
steinf15.txt	10000	156250	2500	3962	5124	3888
steinf16.txt	10000	187500	10	39	38	38
steinf17.txt	10000	187500	20	79	75	75
steinf18.txt	10000	187500	834	1660	2140	1587
steinf19.txt	10000	187500	1250	2227	2909	2161
steinf20.txt	10000	187500	2500	3709	4808	3612

D. Đánh giá kết quả thực nghiệm

Chất lượng lời giải

Mục này nhằm so sánh chất lượng lời giải của các thuật toán SPT-Steiner và PD-Steiner với thuật toán MST-Steiner. Kết quả so sánh được ghi nhận ở bảng 6; bảng này được tổng hợp từ các bảng 1, 2, 3, 4, 5. Nội dung của bảng 6 cho biết số lượng (*SL*) và phần trăm (%) tương ứng với số lượng bộ dữ liệu cho chất lượng lời giải tốt hơn (ghi nhận bởi ký hiệu "<") hoặc cho chất lượng lời giải bằng nhau (ghi nhận bởi ký hiệu "=") hoặc cho chất lượng lời giải tồi hơn (ghi nhận bởi ký hiệu ">") khi so sánh SPT-Steiner và PD -Steiner với thuật toán MST- Steiner.

Với 18 bộ dữ liệu nhóm *steinb*, thuật toán SPT-Steiner cho chất lượng lời giải tốt hơn (tồi hơn) thuật toán MST-Steiner (27.8%, 55.6%); thuật toán PD-Steiner cho chất lượng lời giải tốt hơn (tồi hơn) thuật toán MST-Steiner (66.7%, 0.0%). Kết quả so sánh các thuật toán SPT-Steiner và PD-Steiner với thuật toán MST- Steiner trên mỗi nhóm dữ liệu *steinc, steind, steine, steinf* còn lại cũng đã được thể hiện chi tiết ở bảng 6.

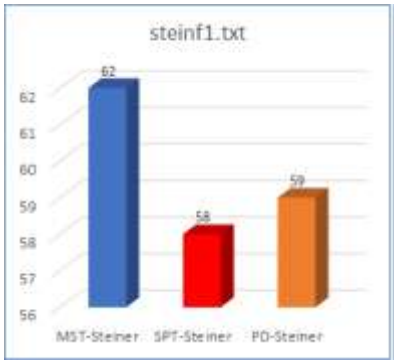
Đánh giá chung trên toàn bộ 98 bộ dữ liệu, thuật toán SPT-Steiner cho chất lượng lời giải tốt hơn (tồi hơn) thuật toán MST-Steiner (26.5%, 66.3%); thuật toán PD-Steiner cho chất lượng lời giải tốt hơn (tồi hơn) thuật toán MST-Steiner (86.7%, 3.1%).

Bảng 6. So sánh chất lượng lời giải của thuật toán SPT-Steiner và PD -Steiner với thuật toán MST- Steiner

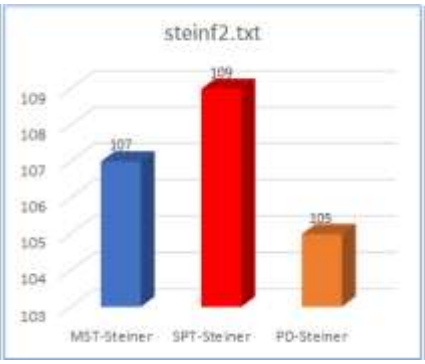
Nhóm đồ thị	SPT < MST		SPT = MST		SPT >MST		PD< MST		PD = MST		PD >MST	
	SL	%	*SL*	%	*SL*	%	*SL*	%	*SL*	%	*SL*	%
steinb	5	27.8	3	16.7	10	55.6	12	66.7	6	33.3	0	0.0
steinc	6	30.0	1	5.0	13	65.0	17	85.0	1	5.0	2	10.0
steind	4	20.0	2	10.0	14	70.0	17	85.0	2	10.0	1	5.0
steine	7	35.0	1	5.0	12	60.0	20	100.0	0	0.0	0	0.0
steinf	4	20.0	0	0.0	16	80.0	19	95.0	1	5.0	0	0.0
Tổng cộng:	26	26.5	7	7.1	65	66.3	85	86.7	10	10.2	3	3.1

Minh họa chất lượng lời giải bằng đồ thị

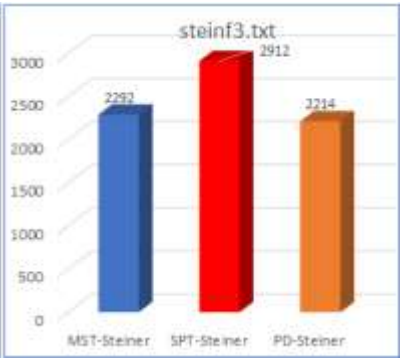
Các hình 6.*a*, 6.*b*, 6.*c*, 6.*d*, 6.*e*, 6*f* sau minh họa chất lượng các thuật toán qua 6 đồ thị đầu tiên thuộc nhóm đồ thị lớn *steinf*; tương ứng với số liệu lấy từ 6 dòng đầu tiên của bảng 5.



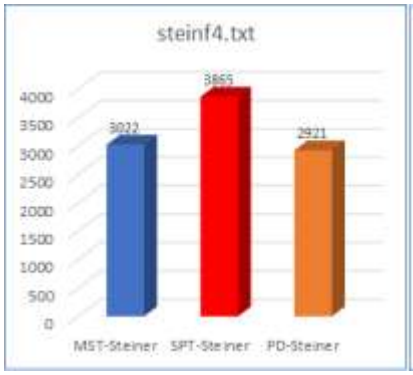
Hình 6.*a*. Minh họa chất lượng *steinf1*



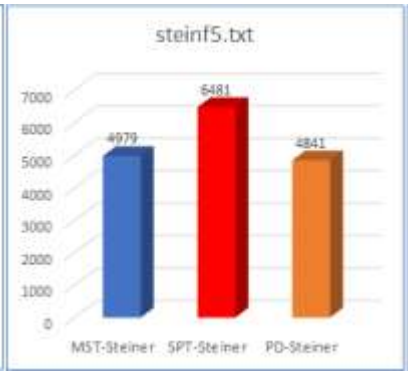
Hình 6.*b*. Minh họa chất lượng *steinf*2



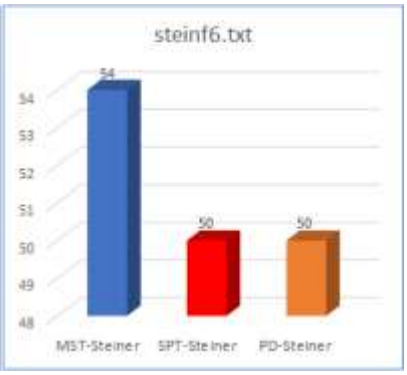
Hình 6.*c*. Minh họa chất lượng *steinf*3



Hình 6.*d*. Minh họa chất lượng *steinf4*



Hình 6.*e*. Minh họa chất lượng *steinf*5



Hình 6.*f*. Minh họa chất lượng *steinf*6

Thời gian chạy của các thuật toán MST-Steiner, SPT-Steiner, PD-Steiner

Tiếp theo chúng tôi ghi nhận thời gian chạy trung bình của ba thuật toán MST-Steiner, SPT-Steiner, PD-Steiner qua các test có số thứ tự 13,14,15 của mỗi nhóm dữ liệu (chúng tôi chọn các test này vì trong mỗi nhóm đồ thị thì các test có số thứ tự 13,14,15 có cùng số đỉnh và có cùng số cạnh).

Thời gian chạy ngoài việc phụ thuộc vào độ phức tạp thời gian tính của thuật toán; còn phụ thuộc vào môi trường thực nghiệm [13],… tuy nhiên thông tin về thời gian chạy của các thuật toán do chúng tôi cài đặt liệt kê ở bảng 7 cũng cho chúng ta thông tin tham khảo sâu sát hơn về ba thuật toán này (chúng tôi chọn cấu trúc dữ liệu hàng đợi ưu tiên để cài đặt ba thuật toán trên).

Bảng 7. Thời gian tính trung bình của các thuật toán theo mỗi nhóm dữ liệu (tính bằng đơn vị giây).

Nhóm đồ thị	MST-Steiner	SPT-Steiner	PD-Steiner
steinb	0.001	0.003	0.001
steinc	0.032	0.298	0.023
steind	0.136	1.752	0.139
steine	0.939	23.334	1.402
steinf	19.594	1241.592	23.393

IV. KẾT LUẬN

Trong bài báo này chúng tôi đã đề xuất hai thuật toán heuristic SPT-Steiner và PD-Steiner giải bài toán *SMT* dựa trên các thuật toán tìm đường đi ngắn nhất; các đề xuất này cùng với thuật toán MST-Steiner của Bang Ye Wu và Kun-Mao Chao đã được chúng tôi cài đặt thực nghiệm và đánh giá chi tiết trên 98 bộ dữ liệu là các đồ thị thưa trong hệ thống dữ liệu thực nghiệm chuẩn; trong đó có những đồ thị thưa có kích thước lớn. Thuật toán SPT-Steiner cho lời giải chất lượng tốt hơn hoặc bằng thuật toán MST-Steiner trên 33.7% số bộ dữ liệu, thuật toán PD-Steiner cho lời giải chất lượng tốt hơn hoặc bằng thuật toán MST-Steiner trên 96.9% số bộ dữ liệu. Thời gian chạy của các thuật toán SPT-Steiner và PD-Steiner chậm hơn so với thuật toán MST-Steiner. Hai thuật toán heuristic SPT-Steiner và PD-Steiner và các kết quả thực nghiệm trong bài báo này là thông tin hữu ích cho các nghiên cứu tiếp theo về bài toán *SMT*; nhất là đối với việc giải bài toán *SMT* trong trường hợp đồ thị thưa có kích thước lớn.

TÀI LIỆU THAM KHẢO

[1]. Ankit Anand, Shruti, Kunwar Ambarish Singh, "An Efficient Approach for Steiner Tree Problem by Genetic Algorithm", International Journal of Computer Science and Engineering, pp.233-237,2015.

[2]. Arie M. C. A. Koster, Xavier Munoz (Eds), "Graphs and Algorithms in Communication Networks", Springer, 2010.

[3]. Bang Ye Wu, Kun-Mao Chao, "Spanning Trees and Optimization Problems", Chapman&Hall/CRC, pp.158-165, 2004.

[4]. Chin Lung Lu, Chuan Yi Tang, Chuan Yi Tang, "The full Steiner tree problem", ELSEVIER, pp.55-67, 2003.

[5]. Celso. C. Ribeiro, Mauricio.C. De Souza, "Tabu Search for the Steiner Problem in Graphs", Networks, Vol.36, pp.138-146, 2000.

[6]. Diego de Una, Graeme Gange, Peter Schachte, Peter J. Stuckey, "Steiner Tree Problems with Side Constraints Using Constraint Programming", Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[7]. David M. Warme, Pawel Winter, Martin Zachariasen, "Exact Algorithms for Plane Steiner Tree Problems: A Computational Study", University of Copenhagen, Denmark, pp.1-34, 1998.

[8]. Ding-Zhu Du, J. M. Smith, J. H. Rubinstein, "Advances in Steiner trees", pp.1-322, 2000.

[9]. Edmund Ihler, Gabriele Reich, Peter Widmayer, "Class Steiner trees and VLSI-design", ELSEVIER, pp.173-194, 1999.

[10]. J. E. Beasley, OR-Library: URL http://people.brunel.ac.uk/~mastjjb/jeb/orlib/steininfo.html.

[11]. J. E. Beasley, "An SST-Based Algorithm for the Steiner Problem in Graphs", Networks, Vol.19, 1-16, l989.

[12]. Jens Vygen, "Steiner Trees in Chip Design", 2009.

[13]. Jack J. Dongarra, "Performance of Various Computers Using Standard Linear Equations Software", 2014.

[14]. Jon William Van Laarhoven, "Exact and heuristic algorithms for the Euclidean Steiner tree problem", University of Iowa, 2010 (Doctoral thesis).

[15]. L. Kou, G. Markowsky, L. Berman, "A Fast Algorithm for Steiner Trees", acta informatica, Vol.15,pp.141-145, 1981.

[16]. Marcello Caleffi, Ian F. Akyildiz, Luigi Paura, "On the Solution of the Steiner Tree NP-Hard Problem via Physarum BioNetwork", IEEE, pp.1092-1106, 2015.

[17]. M. Hauptmann, M. Karpinski (Eds), "A Compendium on Steiner Tree Problems", pp.1-36, 2015.

[18]. Nguyễn Đức Nghĩa, Nguyễn Tô Thành, "Toán rời rạc", NXB Đại học Quốc gia Hà Nội (tái bản lần 8), 2009.

[19]. Nguyen Viet Huy, Nguyen Duc Nghia, "Solving graphical Steiner tree problem using parallel genetic algorithm", RIVF, 2008.

[20]. Robert Sedgewick, Kevin Wayne (Princeton University), "Algorithms", Addison-Wesley, pp.518-701.

[21]. S. L. Martins, M. G. C. Resende, C. C. Ribeiro, P. M. Pardalos, "A parallel grasp for the steiner tree problem in graphs using a hybrid local search strategy", 1999.

[22]. Stefan Hougardy, Jannik Silvanus, Jens Vygen, "Dijkstra meets Steiner: a fast exact goal-oriented Steiner tree algorithm", University of Bonn, 2015.

[23]. S. E. Dreyfus, R. A. Wagner, "The Steiner Problem in Graphs", Networks, Vol.1, pp.195-207, 1971.

[24]. Thomas Bosman, "A Solution Merging Heuristic for the Steiner Problem in Graphs Using Tree Decompositions", VU University Amsterdam, Netherlands, pp.1-12, 2015.

[25]. Trần Quốc Chiến, "Lý thuyết đồ thị và ứng dụng", Nhà xuất bản Đại học Đà Nẵng, 2007.

[26]. Trần Lê Thủy, "Về bài toán Steiner", Viện Toán học, Viện Hàn lâm Khoa học và Công nghệ Việt Nam, 2014 (Luận văn Thạc sĩ).

[27]. Thorsten Koch, Alexander Martin, "Solving Steiner Tree Problems in Graphs to Optimality", Germany, pp.1-31,1996.

[28]. Vũ Đình Hòa, "Bài toán Steiner", http://math.ac.vn.

[29]. Xiuzhen Cheng, Ding-Zhu Du, "Steiner trees in industry", Kluwer Academic Publishers, Vol.5, pp.193-216, 2004.

[30]. Xinhui Wang, "Exact algorithms for the Steiner tree problem", Doctoral thesis, ISSN 1381-3617, 2008.

HEURISTIC ALGORITHMS FOR SOLVING STEINER MINIMAL TREE PROBLEM

Tran Viet Chuong, Phan Tan Quoc, Ha Hai Nam

ABSTRACT: *Steiner Minimal Tree (SMT) is a combinatorial optimization problem that has many important applications in science and engineering; this is the problem of class NP-hard. In recent decades, there have been a series of scientific papers which were published for solving the SMT problem according to the approaches of exact solutions and approximate solutions. In this paper, two heuristic algorithms are proposed for solving the proposed SMT; The first proposal is to use the main idea of the algorithm for finding the shortest path tree and the second one is to combine the main idea of algorithms Prim and Dijkstra. The proposed algorithms have been experimented on the sparse graphs in the standard experimental benchmark instances; Experimental results show that the proposed algorithms are more efficient than some existing approximation algorithms; especially for the sparse graphs with large size. The experimental results are useful information for further research on the problems of SMT.*